Werkzeuge für das wissenschaftliche Arbeiten

Python for Machine Learning and Data Science

Abgabe: 15.12.2023

Inhaltsverzeichnis

1 Projektaufgabe

In dieser Aufgabe beschäftigen wir uns mit Objektorientierung in Python. Der Fokus liegt auf der Implementierung einer Klasse, dabei nutzen wir insbesondere auch Magic Methods.

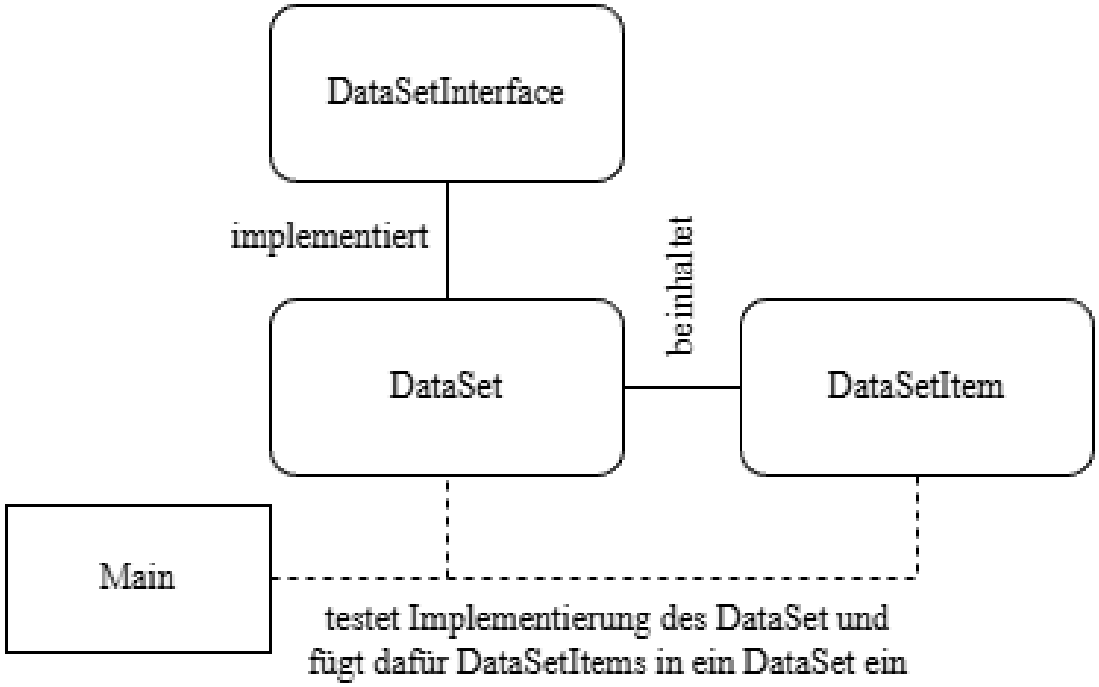

Abbildung 1: Darstellung der Klassenbeziehungen

1.1 Einleitung

Ein Datensatz besteht aus mehreren Daten, ein einzelnes Datum wird durch ein Objekt der Klasse `DataSetItem` repräsentiert. Jedes Datum hat einen Namen (Zeichenkette), eine ID (Zahl) und beliebigen Inhalt.

Nun sollen mehrere Daten, Objekte vom Typ `DataSetItem`, in einem Datensatz zusammengefasst werden. Sie haben sich bereits auf eine Schnittstelle und die benötigten Operationen geeinigt, die ein Datensatz unterstützen muss. Es gibt eine Klasse `DataSetInterface`, die diese Schnittstelle definiert und die Operationen eines Datensatzes vorgibt. Bisher fehlt jedoch die Implementierung eines Datensatzes mit allen Operationen.

Implementieren Sie eine Klasse `DataSet` als Unterklasse von `DataSetInterface`.

1.2 Aufbau

Es gibt drei Dateien: `dataset.py`, `main.py` und `implementation.py`[1]

In der Datei `dataset.py` befinden sich die Klassen `DataSetInterface` und `DataSetItem`. In der Datei `implementation.py` muss die Klasse `DataSet` implementiert werden. Die Datei `main.py` nutzt die Klassen `DataSet` und `DataSetItem` und testet die Schnittstelle und deren Operationen.

1.3 Methoden

Bei der Klasse `DataSet` sind insbesondere die folgenden Methoden zu implementieren. Die genaue Spezifikation finden Sie in der Datei `dataset.py`.

1. `__setitem__(self, name, id_content)` Hinzufügen eines Datums mit Name, ID und Inhalt
2. `__iadd__(self, item)` Hinzufügen eines `DataSetItem`
3. `__delitem__(self, name)` Löschen eines Datums auf Basis des Namens. Der Name ist dabei ein eindeutiger Schlüssel und darf pro Datensatz nur einmal vorkommen.
4. `__contains__(self, name)` Prüfung, ob ein Datum mit diesem Namen im Datensatz vorhanden ist
5. `__getitem__(self, name)` Abrufen eines Datums über seinen Namen
6. `__and__(self, dataset)` Bestimmung der Schnittmenge zweier Datensätze
7. `__or__(self, dataset)` Bestimmung der Vereinigung zweier Datensätze
8. `__iter__(self)` Iteration über alle Daten eines Datensatzes
9. `__filtered_iterate__(self, filter)` Gefilterte Iteration über einen Datensatz anhand einer Lambda-Funktion
10. `__len__(self)` Rückgabe der Anzahl der enthaltenen Daten

[1] Die Dateien befinden sich im Ordner `/code/` dieses Git-Repositories.

2 Abgabe

Programmieren Sie die Klasse `DataSet` in der Datei `implementation.py` zur Lösung der oben beschriebenen Aufgabe im VPL.

Sie können die Entwicklung auch lokal durchführen. Die benötigten Dateien stehen dazu im Moodle zum Download bereit.

Das VPL nutzt den gleichen Code, wobei die Datei `main.py` um zusätzliche Testfälle und Überprüfungen erweitert wurde, um sicherzustellen, dass die korrekten Klassen verwendet werden.